OULUN AMMATTIKORKEAKOULU

Petri Luoma

PILVIPALVELUN KÄYTTÖLIITTYMÄDEMO PROJEKTINHALLINNAN DOKUMENTOINTIIN

PILVIPALVELUN KÄYTTÖLIITTYMÄDEMO PROJEKTINHALLINNAN DOKUMENTOINTIIN

Petri Luoma
Opinnäytetyö
Kevät 2014
Tietojenkäsittelyn koulutusohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma

Tekijä(t): Petri Luoma
Opinnäytetyön nimi: Pilvipalvelun käyttöliittymädemo projektinhallinnan dokumentointiin
Työn ohjaaja(t): Marja-Leena Korva
Työn valmistumislukukausi ja -vuosi: kevät 2014 Sivumäärä: 31

Opinnäytetyönä suunniteltiin ja toteutettiin käyttöliittymän demo projektinhallinnan dokumentoinnin hallintaan ja ylläpitoon tarkoitettuun pilvipalveluun. Työn pohjana toimi aiemmin tehty versio samasta ohjelmasta, joka siirrettäisiin Internet-selaimessa toimivaksi pilvisovellukseksi. Työ toteutettiin käyttämällä HTML5-merkkauskieltä ja sen ohessa toimivia skriptauskieliä Javascriptiä ja jQueryä sekä uutta CSS3-tyylinmuokkauskieltä. Tuotteen vaatimuksia olivat helppokäyttöisyys ja selkeys, jotta tuotteen käyttöönottokynnys olisi mahdollisimman pieni.

Työn suunnittelussa huomioitiin mobiilien laitteiden yleistyminen ja Adobe Flashin markkina-aseman pieneneminen. Näiden avulla päätettiin, mitä toteutustapoja työssä lopulta käytettiin. Demon vaatimuksena oli saada esiteltyä mahdollisimman viimeistelty tuote, joka vastaisi ulkonäöltään ja toiminnallisuudeltaan mahdollisimman paljon lopullista tuotetta. Toisena tavoitteena oli mahdollistaa demoon käytetyn työn ja koodin käyttö lopullisessa tuotteessa mahdollisimman laajasti, mahdollisimman pienten muutosten jälkeen.

Työn toimeksiantajana toimi raahelainen tietohallinnon palveluja tarjoava yritys Enfide Ky. Tietoperustana käytettiin koulussa eri kursseilta opittuja perusteita sekä omatoimista tutkintaa eri tilanteiden vaatimusten mukaan. Aineistona käytettiin Internetiä trendien ennustuksia katsottaessa, sekä yritykseltä löytynyttä kirjallisuutta HTML5:den perusperiaatteista ja käytöstä.

Työn tuloksena syntyi toimiva käyttöliittymä, jonka perusominaisuudet saatiin pienten muutosten jälkeen siirrettyä valmiiseen tuotteeseen. Työn perustana toiminut tutkimus antoi myös tietoa HTML5:n eri vahvuuksista ja heikkouksista ja yleisesti käyttöliittymän suunnittelusta. Tuotteen kehitys jatkui demon valmistumisen jälkeen lopullisen tuotteen kehityksenä, ja tuotteeseen lisättiin palvelinpohjaista toiminnallisuutta, sekä ulkoasun parannuksia.

Asiasanat: HTML5, jQuery, Pilvipalvelu, käyttöliittymä

ABSTRACT

Oulu University of Applied Sciences
Degree programme of Business Information Systems

Author(s): Petri Luoma
Title of thesis: Designing and Implementing a User Interface Demo for a Cloud Application
Supervisor(s): Marja-Leena Korva
Term and year when the thesis was submitted: Spring 2014 Number of pages: 31

The goal of the thesis was to design and produce a functioning user interface demo for a cloud application supporting project management documentation. An earlier version of the software functioned as the basis of the new application which was converted into a cloud application. The application was made using the HTML5 mark-up language, and accompanying scripting languages, Javascript and jQuery, and CSS3 style markup language. The main emphasis for the software was its ease-of-use and clarity, so that its deployment threshold would be as low as possible.

Trends of different browser usage and implementation methods were also taken into consideration while designing the product. The starting point for the research was the fact that HTML5 becoming more popular compared to Adobe Flash, and mobile platforms are becoming more common among the end users. Consequently, this influenced the selection of implementation methods used in this project. The main requirement for the demo was to make the demo look and feel as close to the final product as possible, while trying to make as much of the code and other resources reusable in the final product with as few changes as possible. The assigner of the project was an ICT-service providing company Enfide Ky, located in Raahe Finland. The team used their basic knowledge from university courses and independent research as the basis of the design and development of the product. The research material used was mostly assorted literature on HTML5 basics, and various sites on the Internet on different trend predictions and other assorted knowledge.

The main result of the project was a functioning demo that the company used to sell the product to customers and most of the code was able to be imported into the final product. The base research made for the project gave the team a lot of experience in working with HTML5 and designing user interfaces. The development of the product continued after the demo was finished and the product was refactored into an actual cloud based service with database functionality and improved look of the user interface.

Keywords: HTML5, jQuery, Cloud Service, User Interface

LYHENTEET

Asp.net	Microsoftin kehittämä Web-ohjelmistokehys
CSS	Cascading style sheet. Määrää sivun ulkoasun ominaisuudet kuten esimerkiksi taustavärin ja fontin.
CSS3	Uusin CSS-versio. Sisältää mm. animaatioita
Django	Python kirjasto, joka hoitaa palvelinpuolen ja tietokannan toiminnot.
DOM	Document object model on ohjelmointikielestä ja -alustasta riippumaton tapa, jolla ohjelma toimii vuorovaikutuksessa HTML-dokumentin kanssa.
DRM	Digital Rights Management, tekijänoikeuksien suojaaminen
Git Bash	Ohjelma, joka hoitaa lähdetiedostojen hallinnan muokkaamalla vain niitä rivejä, jotka eroavat edellisestä versiosta.
Google Docs	Google pilvessä toimiva dokumentin muokkausohjelma.
HTML	Hypertext markup language. Määrittää sivun sisällön
HTML5	Uusin html:n versio. Sisältää mm. drag and drop ominaisuuden sekä tiedostojen upload yms. Toimintoja
IAAS	Infrastructure As A Service, joka tarjoaa lähinnä tietokannan ja serverit käyttäjälle.
Javascript	Määrittää sivun toiminnallisuuden/interaktiivisuuden
jQuery	Javascript kirjasto. Helpottaa tiettyjen asioiden tekoa javascriptin avulla.
Mock-up	Staattinen piirros, jonka tarkoituksena on saada nopeasti näkyvä versio jostain asiasta. Yleensä toteutettu käsin piirtämällä tai photoshopilla.
PAAS	Platform As A Service. Pilvipalvelun tyyppi joka tarjoaa kokonaisen ohjelmisto- ja laitteistoinfrastruktuurin käyttäjälle, sisältäen esimerkiksi käyttöjärjestelmän.
PHP	Palvelinpuolen ohjelmointikieli
Pilvipalvelu	Ohjelmistopalvelu joka pyörii jossain palveluntarjoajan määräämällä koneella/serverillä täysin itsenäisesti ilman että loppukäyttäjän tarvitsee tehdä mitään ohjelmiston huollon tai päivittämisen kanssa.
Python	Skriptauskieli
Razor	Asp.net view engine
Requirement management	Ohjelmiston/Projektin vaatimuksien määrittely ja niiden etenemisen/sisällön seurantaa ja luomista

SAAS	Software As A Service. Pilvipalvelutyyppi joka tarjoaa ohjelmiston joka pyörii jossain palveluntarjoajan palvelimella.
Selain	Ohjelma, joka tulkitsee HTML:n, CSS:n ja Javascriptin käyttäjälle muodostaen nettisivun.
View Engine	Moduuli joka mahdollistaa uudenlaisia syntaksi mahdollisuuksia
W3C	World Wide Web Consortium. Kansainvälinen yhtymä, joka valvoo web-kielten kehitystä.

SISÄLLYS

1 JOHDANTO

Työn tarkoituksena oli tehdä käyttöliittymädemo Enfide-nimisen yrityksen tulevalle pilvipalvelulle, jonka avulla tuotteen käyttäjä voisi tehdä ja muokata ohjelmistomäärittelyn projektinhallintaan liittyviä dokumentteja. Tuotteeseen sisältyisi myös kommentointimahdollisuus sekä mahdollisuus tallentaa erilaisia pohjia tehdyistä dokumenteista tai määrittelyistä. Tavoitteeksi asetettiin myös, että mahdollisimman suurta osaa koodista voisi käyttää lopullisessa tuotteessa minimaalisin muutoksin. Yritys oli jo päättänyt, että ohjelma tehdään HTML5:llä, mutta muuten kehitystiimi sai vapaat kädet päättää ohjelmiston toteutustavoista ja käyttöliittymästä.

Lopullinen työ tulisi toimimaan jonkinlaisena pilvisovelluksena, koska ohjelman haluttiin olevan mahdollisimman mobiili sekä ohjelmistoriippumaton. Pilvipalvelun toteutustapaa ja muotoa mietittiin vasta itse tuotetta kehitettäessä, minkä takia palvelinpohjainen toiminnallisuus minimoitiin demossa.

2 TYÖN TEOREETTINEN VIITEKEHYS

2.1 Käyttöliittymä

Käyttöliittymä on tapa, jolla käyttäjä on vuorovaikutuksessa ohjelmiston kanssa. Hyvän käyttöliittymän vaatimukset vaihtelevat tuotteittain, ja riippuvat paljolti siitä, mitä ominaisuuksia kehittäjät haluavat painottaa. Yleisiä hyvän käyttöliittymän ominaisuuksia on monia. Applen käyttöliittymän perusteet sisältävässä dokumentissa (Apple 1992, hakupäivä 11.5.2014) listataan muutamia näistä. Samoja perusperiaatteita käytettiin tehdyssä sovelluksessa, lähinnä keskittyen tuotteen helppokäyttöisyyteen ja selkeyteen, sillä yhtenä kehitystavoitteena oli, että jopa henkilöt jotka eivät normaalisti käytä paljon tietokonetta, voisivat tehdä vaatimuksen määrittelyjä helposti.

2.2 Projektinhallinnan dokumentointi

Projekteja suunniteltaessa on hyvä kirjata kaikki päätetyt ominaisuudet, toteutustavat, rajapinnat ynnä muut sellaiset, erilaisiin dokumentteihin. Näiden dokumenttien avulla oman projektin ominaisuudet ja tarpeet pysyvät hyvin hallinnassa, ja projektin ominaisuuksien ja muiden tietojen jakaminen muille on helppoa. Myös itse toteutus nopeutuu huomattavasti, sillä ohjelmoijien ei tarvitse miettiä laajan mittakaavan ratkaisuja ja seurauksia koodatessaan tiettyä ominaisuutta. (Department for Business Innovation & Skills 2010, hakupäivä 11.5.2014). Muita vastaavia tuotteita ei tarkasteltu. Osittain syynä oli se, ettei vastaavista tuotteista ollut ilmaista versiota.

2.3 Pilvipalvelut yleisesti

Pilvipalveluilla tarkoitetaan ohjelmistoa, joka toimii jossain palvelun tarjoajan palvelimella, ja on täysin erillään käyttäjän laitteesta. Pilvipalvelut lajitellaan yleensä kolmeen kategoriaan, PAAS, SAAS ja IAAS. Työn kohteena ollut sovellus suunniteltiin aluksi PAAS-tyyliseksi pilvipalveluksi, mutta lopulta se kehittyi SAAS-palveluksi. SAAS-lyhenne tulee sanoista Software As A Service, ja se tarkoittaa sitä, että pilvisovelluksen tarjoaja huolehtii ohjelmiston, sekä mahdollisten servereiden pystytyksestä, ylläpidosta ynnä muusta. Hyvä esimerkki SAAS-palvelusta on Google Apps. Muut pilvipalvelun tyypit, IAAS (Infrastucture As A Service) ja PAAS (Platform As A

Service), olivat tarpeettoman laajoja työn tarpeisiin nähden. IAAS on palvelutyyppi, jossa asiakkaalle tarjotaan mahdollisuudet oman palvelun ylläpitoon, ja PAAS on laajempi kokonaisuus, joka tarjoaa kokonaisen käyttöjärjestelmän, ohjelmointikielen ympäristön ynnä muita vastaavia ominaisuuksia palvelimen ja tietokannan lisäksi. (United States Computer Emergency Readiness Team 2011, hakupäivä 11.5.2014).

2.4 HTML5 yleisesti

Uusi HTML5 tuo monia uusia ominaisuuksia verrattuna vanhempiin HTML-versioihin. HTML5:n ydin-uudistuksiin kuuluu uusimpien multimedia- tyyppien tukeminen, pitäen samalla koodien luettavuuden yksinkertaisena ja helposti luettava, sekä ihmisten, että selainten näkökulmasta. HTML5 lisää myös uusia tageja. Multimedia ominaisuuksien lisäämistä helpottaa esimerkiksi canvas-tagi, johon käyttäjä voi piirtää grafiikkaa. Syntaksia selkeyttää esimerkiksi section- tai article-tagit. Myös muutamia vanhoja tageja on muokattu tai standardoitu ja dokumenttien syntaksivirheiden käsittely on standardoitu. HTML5:n mukana tulee myös uudistuksia HTML-dokumenttien tyylin muokkaukseen CSS3:n avulla. HTML5:den määrityksiä voi lukea W3C:n virallisilta HTML5-sivulta (W3C 2014, hakupäivä 11.5.2014).

2.5 Muut ohjelmat yleisesti

Ohjelmistonkehitysympäristönä käytettiin Microsoftin WebMatrix–ohjelmaa. Verkkosivun julkaiseminen oli erittäin helppoa WebMatrixin avulla ja siihen sisältyi Microsoftin IIS-palvelin. Muilta osin ohjelma oli huono: esimerkiksi tiedostojen scrollaaminen hiiren rullaa käyttämällä lakkasi toimimasta erään päivityksen jälkeen.

2.5.1 Skriptauskielet

jQuery on Javasriptillä toteutettu kirjasto, jonka tarkoituksena on helpottaa eri Javascript ominaisuuksien toteutusta. jQueryUI laajentaa vielä jQueryn perusominaisuuksia ja tuo valmiita käyttöliittymän osa-alueita, kuten ikoneja ja tiettyjä valmiita animaatioita suoraan käytettäväksi. Kehityksessä jQueryä käytettiin lähinnä käyttöliittymän ominaisuuksiin, kuten valikkojen piilottamiseen ja työkaluvinkkien näyttämiseen. Myös jQuery UI:n tuomia valmiita animaatioita, sekä sen valmiita ikoneita käytettiin toteutuksessa laajalti. Tiettyjen ominaisuuksien tyylejä jouduttiin kuitenkin muokkaamaan jQueryUI:n lähdekoodissa, sillä jQuery UI:n metodit tuovat mukana ennalta määritellyn näköisiä palkkeja, ynnä muita sellaisia sivun elementtejä, mitkä eivät soveltuneet tuotteen suunniteltuun ulkoasuun.

2.5.2 Razor

Razor on Microsoftin kehittämän asp.net:in osa-alue ja se perustuu C#-ohjelmointikieleen. Razorin perusperiaatteena oli templaattipohjainen ohjelmistokehitys, jonka takia se soveltui hyvin omaan projektiimme. Razorin omia ominaisuuksia ei käytetty oikeastaan ollenkaan, poikkeuksena tiedostoja lataaminen dokumentteihin sisälle, mutta tästä huolimatta Razorin käyttö helpotti ohjelmiston päivitystä ja vanhojen ominaisuuksien muokkausta.

3 SOVELLUKSEN SUUNNITTELU

Sovellus suunniteltiin pilvipalveluksi, jolloin käyttäjä voisi käyttää sovellusta laitteistosta riippumatta. Tämä ei juuri koskenut demon tekoa, sillä se ei käyttänyt tietokantaa hyväksi, mutta lopulliseen versioon pilvipalvelun tyypin ja palveluntarjoajan valinta oli erittäin tärkeä valinta. Myös sovelluksen toiminnan varmistaminen eri laitteilla ja etenkin eri selaimilla oli erittäin tärkeä, varsinkin ottaen huomioon mobiilien laitteiden aseman vahvistuminen Internetin selailussa (StatCounter, Hakupäivä 11.5.2014).

Tietyt toiminnallisuudet toimivat eri tavalla tietokoneissa ja kännyköissä, esimerkiksi sovelluksen avainasemassa oleva ominaisuus Drag And Drop. Myös tiettyjä ominaisuuksia suunniteltiin toteutettavaksi mobiililaitteiden takia. Esimerkiksi offline-tilassa työskentely on hyvä ominaisuus liikkuviin järjestelmiin. Käyttäjä voisi esimerkiksi työskennellä määrittelyn parissa lennon aikana, ja tallentaa muutokset pilveen päästessään takaisin langattoman verkon alueelle.

3.1 HTML5:n valinta

Yritys oli valinnut jo etukäteen, että sovellus toteutetaan HTML5-tekniikalla. Koska HTML 5 tukee vanhoja HTML:n ominaisuuksia, HTML 5-version valinta oli helppo päätös. HTML:n rinnalle pohdittiin Adoben Flashiä, jonka avulla käyttöliittymän toiminnallisuus saataisiin toteutettua. Flashin etuna oli se, että se toimii samalla tavalla selaimesta riippumatta, sekä se että siinä on ominaisuuksia joita HTML5:ssä ei ole, esimerkiksi DRM ja webbikameran käyttö. Muilta ominaisuuksilta HTML5 oli selvästi parempi valinta tässä tapauksessa. HTML5 on myös avoin standardi ja sen käyttö on ilmaista, verraten Flashiin, joka on Adoben omistama ja maksullinen (W3C 2014, hakupäivä 11.5.2014). Lisäksi vaikka HTML5:n ominaisuuksien tuki vaihtelee eri selaimilla, Flash ei toimi ollenkaan Applen tuotteissa ja koska palvelu tulisi toimimaan pilvessä, oli erittäin tärkeää saada sovellus toimimaan mahdollisimman monella eri laitteella (Jobs S. April 2010, hakupäivä 11.5.2014).

Tuotteen kehitystä aloitettaessa Flash oli yleisemmin käytössä HTML5:teen verrattuna, mutta tilanne on jo hieman muuttumassa. Myös aikaisempi kokemus HTML-stä sekä Javascriptistä, jotka ovat suurin piirtein muuttumattomia uusissa versioissaan verrattuna koulussa opittuihin

versioihin, vaikutti toteutustavan valintaan. Lopulta kuitenkin se, että pilvipalvelun pääperiaate on laitteistoriippumattomuus, viimeisteli HTML5:n valinnan.

.

3.2 Työn suunnittelu

HTML5 oli valittu työn toteutuskieleksi jo projektin suunnittelun alkuvaiheessa siitä huolimatta, että tiettyjen HTML5:n ominaisuuksilta puuttui täysi selaintuki. (CanIUse 2014, hakupäivä 11.5.2014) HTML5 oli kuitenkin aktiivisessa kehityksessä, eli puuttuvat ominaisuudet lisätään selaimiin päivitysten muodossa, jolloin ominaisuuksien puuttuminen oli vain väliaikaista. Toteutustavaksi valittiin HTML5, johtuen sen uusista ominaisuuksista joita hyödynnettäisiin ohjelman toteutuksessa. Monet ohjelman ominaisuuksista olisi mahdollista, ja osa jopa parempi, toteuttaa ilman HTML5:ttä, mutta kehityskieleiksi valittiin silti HTML5 johtuen sen kokoajan etenevästä kehityksestä.

3.2.1 Skriptauskielen valinta

jQuery-kirjasto valittiin tukemaan Javascriptiä, sillä se helpottaa monien ominaisuuksien lisäämistä ja varmistaa ko. ominaisuuden toiminnallisuuden useissa selaimissa ja tukee kaikkia yleisimpiä selaimia (jQuery 2014, hakupäivä 11.5.2014). Työssä käytettiin myös jQueryn laajennusta jQueryUI;tä, joka helpottaa monien käyttöliittymän toiminnallisuuden ja ulkoasun piristämisen ominaisuuksia, kuten esimerkiksi elementtien animointia. Kehitysryhmällä oli myös kokemusta kyseisistä skripkirjastoista, joten niiden käyttöönotto oli nopeaa ja helppoa.

3.2.2 Backendin suunnittelu

PHP:n käyttö suljettiin pois pienten kokeilujen jälkeen, sillä kummallakaan ohjelmoijista ei ollut paljoa kokemusta PHP:n käytöstä. Tämän takia koko demo suunniteltiin siten, ettei se tarvitsisi tietokanta-pohjaista toiminnallisuutta. Tämä oli mahdollista, koska kyseessä oli demo joka esiteltiin asiakkaille, ilman että he käyttäisivät itse tuotetta. Näin ollen kaikki tieto, jota ohjelmassa näytettiin, voitiin tallentaa valmiiksi sovelluksen lähdekoodiin ja asiakkaille voitiin selittää tietojen katoaminen sivulla navigoidessa.

3.2.3 Tietoturva

Koska sovellusta käytettiin kontrolloiduissa esittelyissä, eli yrityksen oman työntekijän esitellessä tuotetta asiakkaalle, tietoturva päätettiin jättää matalalle prioriteetille. Myös tiimin oma kokemattomuus tietoturva-asioissa olisi hidastanut ohjelmiston kehitystä, eikä työn lopputulos olisi ollut riittävän laadukas.

3.3 Ohjelman ydinominaisuudet

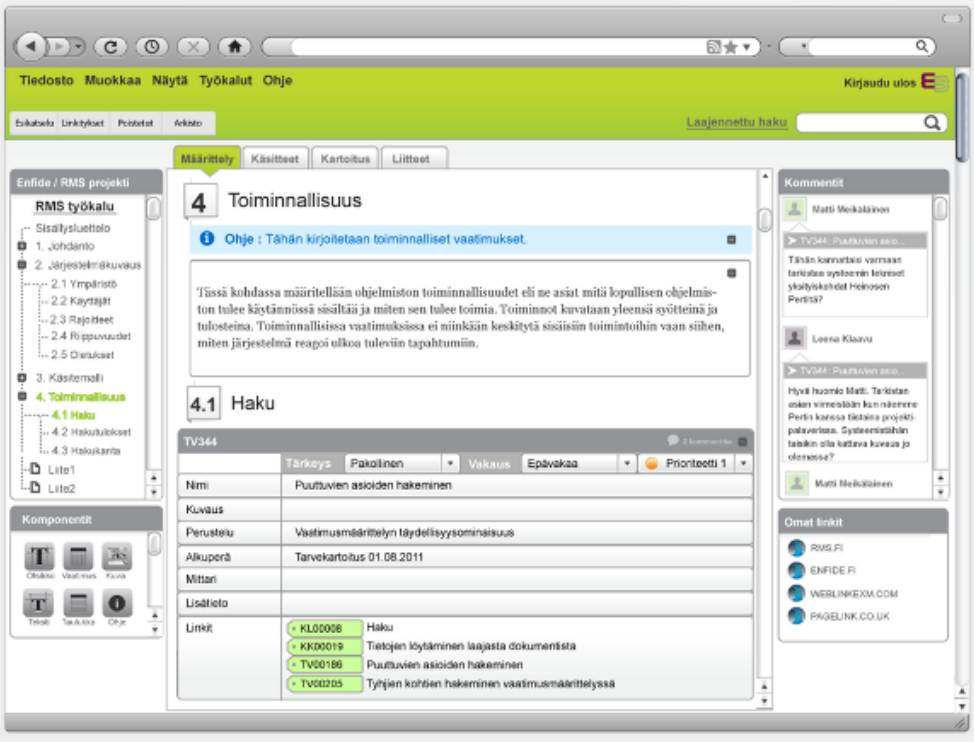
Ohjelman tarkoituksena oli helpottaa projektinhallinnassa käytettävien dokumenttien hallintaa ja ylläpitoa. Tätä silmälläpitäen ohjelmaan suunniteltiin erilaisia tukiominaisuuksia, jotka olisivat avainasemassa ohjelmiston käyttöprosessissa. Ensimmäinen ominaisuus oli kommunikointityökalun, jonka avulla projektissa mukana olevat henkilöt voisivat lähettää erilaisia viestejä, joko komponenttikohtaisesti, dokumenttikohtaisesti tai projektikohtaisesti, kaikille projektissa mukana oleville henkilöille. Kommunikointimahdollisuus lisäsi joustavuutta dokumentinhallintaan, sillä dokumenttien virheiden korjaaminen tai muuttaminen helpottui useamman ihmisen yhteistyön kautta. Toinen avainasemassa oleva ominaisuus oli komponenttien linkitys toisiinsa. Tällä tarkoitettiin lähinnä sitä, että jos kaksi komponenttia ovat vuorovaikutuksessa toistensa kanssa, käyttäjälle kerrottiin tästä, jolloin käyttäjän ei tarvitse itse miettiä, mihin tämän tekemä muutos vaikuttaa.

3.4 Käyttöliittymän ominaisuuksia

Tuotteen sisältö koostui erilaisista komponenteista, esimerkiksi vaatimuskomponentti, tekstikomponentti tai kuvakomponentti, joita käyttäjä raahaisi dokumenttiin. Demon tavoitteena oli luoda kuva siitä, miten lopullisen tuotteen käyttö tulisi toimimaan ja miltä työn ulkoasu näyttäisi. Yksi ensimmäisistä suunnitelluista ominaisuuksista oli komponenttien raahaaminen listasta käyttäen HTML5:n uutta drag and drop ominaisuutta. Tämän päätettiin olevan luontevin ja modernein tapa toteuttaa uusien komponenttien lisääminen dokumenttiin. Sen avulla komponentti voitaisiin laittaa oikeaan kohtaan, kun tuplaklikkaamalla komponentti siirtyisi dokumentin loppuun. Drag and drop toimii myös hyvin kosketusnäytön kanssa pienten skriptin muutosten jälkeen.

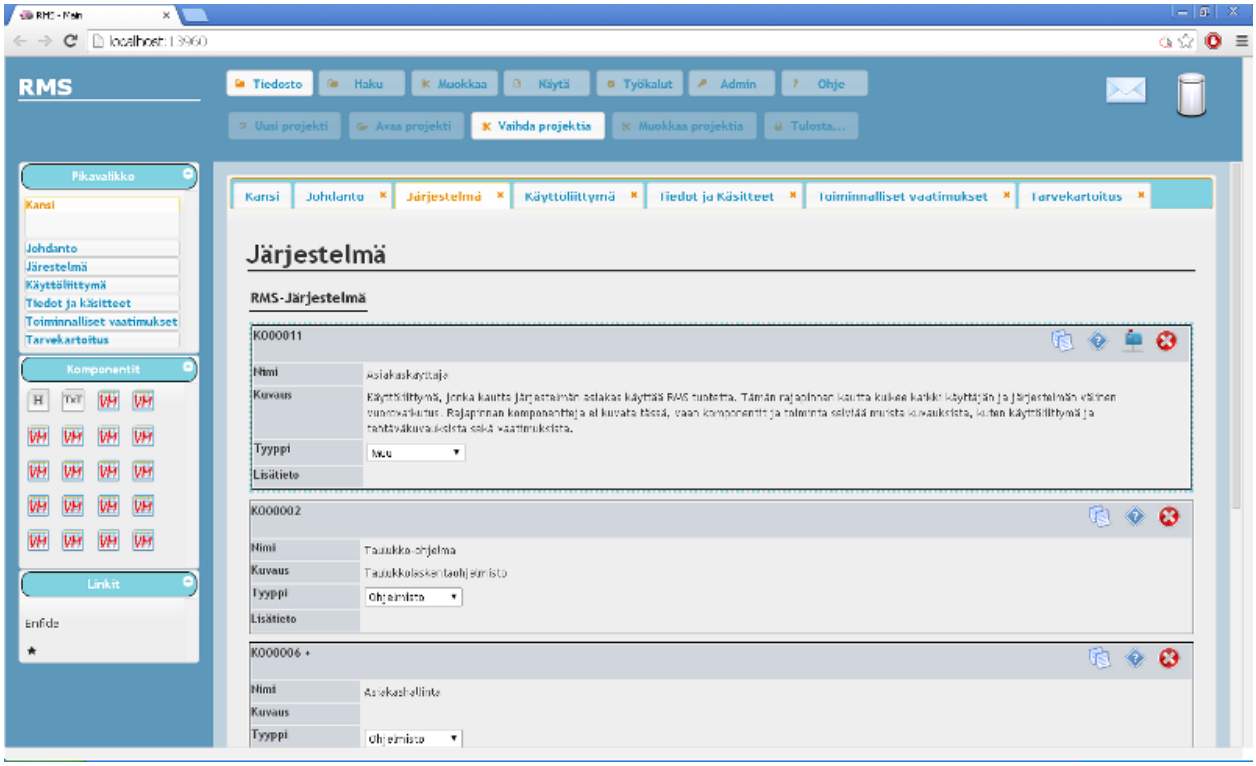
3.4.1 Komponenttien suunnittelu

Toinen perusominaisuus oli itse komponenttien täyttäminen. Koska suuri osa dokumentin teon ajasta käytetään täyttämällä komponentteja, kirjoitustilan on oltava iso ja selkeä. Tämän saavuttamiseksi säilyttäen samalla muiden ominaisuuksien nopeakäyttöisyys, kaikki työkaluvalikot tehtiin minimoitaviksi. Komponenttien työkaluvalikot olivat minimoituja oletuksena, sillä komponenttien työkalujen käyttö oli vielä tässä vaiheessa harvempaa, esimerkkinä vaikka komponenttien kopiointi. Käyttöliittymän ulkoasun pohjana käytettiin yrityksen valmiiksi teettämää mock-uppia, jonka värimaailmaa muokattiin vastaamaan yrityksen värejä. Kuviossa 1 näkyy ohjelman päänäkymä käyttäjän avatessa projektin dokumentin.



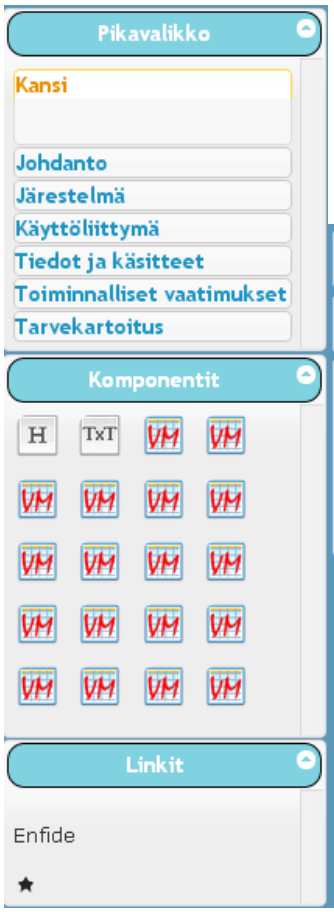
KUVIO 1: Alkuperäinen Mock-Up

Kuviossa 2 on sama näkymä demon käyttöliittymästä tiedosto-ylävalikon ollessa auki.



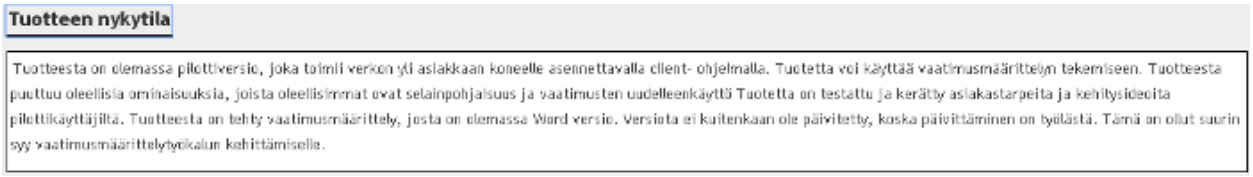
KUVIO 1: Demon lopullinen käyttöliittymä

Kuviossa 3 ja 4 näkyvät sivupaneelin pika-, komponentti- ja linkitvalikko. Pikavalikko toimi dokumentin sisällysluettelona, jonka otsikoita klikkaamalla käyttäjä siirtyi kyseisen otsikon luo päänäkymässä. Komponentit-valikko sisältää kaikki komponentit joita käyttäjä voi lisätä dokumenttiin, esimerkkinä kaksi ensimmäistä komponenttia otsikko- ja tekstikomponentti. Komponenttien ikonit eivät olleet valmiita demoa tehdessä, johtuen niiden suunnittelun haasteellisuudesta. Linkit-valikko oli tarkoitettu käyttäjän dokumenttia koskeviin linkkeihin, ohjelman ulkoisiin tai sisäisiin, joita käyttäjä pystyi lisäämään painamalla tähti-ikonia.. Linkit-valikko poistettiin lopullisesta versiosta. Samat valikot ovat minimoituina kuviossa 4 ja laajennettuina kuviossa 3.

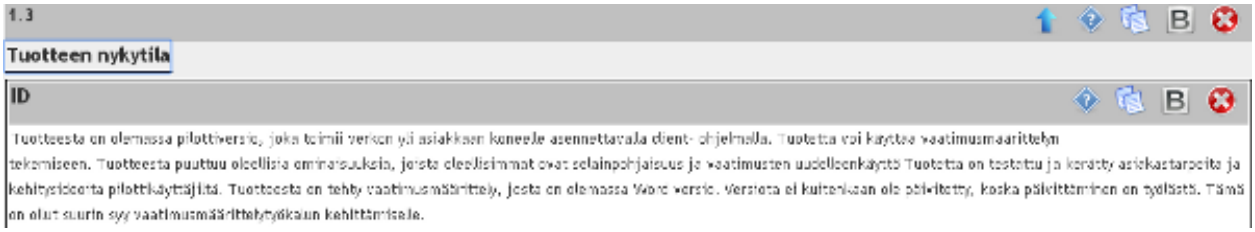


KUVIO 2: Valikot auki ***KUVIO 3: Valikot Kiinni***

Kuviot 5 ja 6 näyttävät otsikko- tekstikomponentin ulkonäön ja toiminnallisuuden. Käyttäjä kykeni kirjoittamaan tekstiä normaalin tekstinmuokkaus ohjelman tapaan, ja muokkaamaan sitä rajoitetusti komponentin ylävalikon ikoneilla. Ikonit ovat oletuksena piilotettuja, mutta tulevat esille komponenttia tuplaklikkaamalla. Lopulliseen versioon ikonit olivat oletuksena esillä, ja piiloutuivat tuplaklikkaamalla. Ylempi komponentti on otsikkokomponentti, Tekstillä ”Tuotteen Nykytila” ja sen ikonit toimivat seuraavasti. Ensimmäinen ikoni on ainoa toiminto, jota ei ole tekstikomponentissa ja se nostaa otsikon tasoa ylemmäksi. Seuraavat ikonit ovat ohjeistus, komponentin kopiointi sisältöineen, tekstin lihavointi ja komponentin poisto.



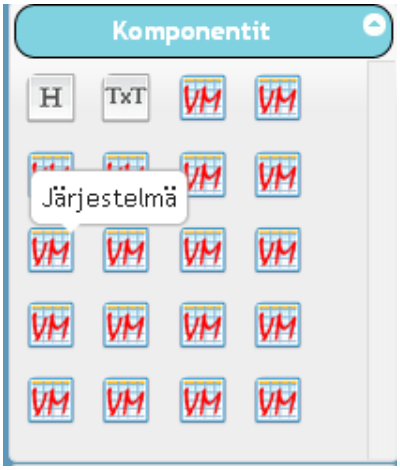
KUVIO 4: Komponentin valikko kiinni



KUVIO 5: Komponenttien valikot auki

3.4.2 Helppokäyttöisyys

Eräs ohjelman päätavoitteista oli se, että ohjelman käyttö olisi mahdollisimman helppoa. Ikonit pyrittiin pitämään mahdollisimman selkeänä ja kuvaavina, mikä osoittautui todella haastavaksi tiettyjen komponenttien kanssa. Lopulta demoon tehtiin vain yksi ikoni vaatimusmäärittely-komponentille, jota käytettiin placeholderina. Lopullisen version komponenttivalikon kuvakkeisiin laitettiin myös komponentin nimen, joka oli minimoitavissa, kun käyttäjä oppi yhdistämään komponentin ikonit niiden nimiin. Myös työkaluohjeiden käyttö oli erittäin tärkeä ja tehokas tapa kertoa käyttäjälle eri ikoneiden toiminnallisuudesta ja tarkoituksesta. Kuviossa 7 näkyy Järjestelmä-komponentin työkaluvinkki.



KUVIO 6: Työkaluvinkki esimerkki

3.4.3 Värisokeus

Käyttöliittymää suunniteltaessa otettiin myös huomioon värisokeat käyttäjät. Tätä pidettiin hyvänä lisäominaisuutena, sillä noin 8 % miehistä ja 0.5 % naisista, on värisokeita. Käyttöliittymän suunnittelussa tämä tarkoittaa sitä, että värien käyttöä täytyy miettiä tarkkaan, ja eri elementtien erottamista toisista useammalla eri tavalla, pelkän värin korostamisen sijaan. Sen sijaan että

elementtien korostaminen tehtäisiin eri värien samalla tummuudella, elementin väritummuutta lisätään tai lasketaan. Käytännössä siis sen sijaan että elementti vaihdettaisiin tummanvihreästä tummansiniseksi, elementti vaihdetaan tummanvihreästä vaaleanvihreäksi. Myös elementin koon ja sijainnin muunnosta voi ajatella korostamaan tiettyjä asioita. Sen sijaan, että dokumentin aktiivinen välilehti olisi erivärinen muihin verrattuna, sen sijaintia voisi laskea tai nostaa muutamalla pikselillä (Johnson J. 2010, hakupäivä 11.5.2014).

4 KEHITYSTEHTÄVÄN HAASTEET

4.1 HTML5 toteutuksen rajoitukset

Koska sovellus toteutettiin HTML5:llä, joka on suhteellisen uutta teknologiaa, tietyt HTML5:n ominaisuudet toimivat eri tavoin, tai eivät ollenkaan, eri selaimissa (Irish P. & Manian D 2013, Hakupäivä 11.5.2014). Esimerkiksi komponenttien lisäämiseen ja siirtelyyn käytetty drag-and-drop toimi ainoastaan Google Chrome -selaimessa. Tämän takia päätettiin, että sovelluksen demotilanne tehdään aina Googlen Chrome -selaimessa, johtuen Chromen parhaimmasta HTML5 tuesta. Myös tietyt ominaisuudet tulisivat mahdollisesti toimimaan eri tavoin erilaisissa laitteissa, etenkin drag and drop ominaisuuden käyttäminen kosketusnäytöllisissä laitteissa. Myös tyylitiedoston osa-alueet toimivat eritavoin eri selaimissa. Firefox käyttää Gecko layout engineä, kun taas Chrome ja Opera käyttävät WebKittiä. Microsoftilla on myös heidän oma layoutenginensä, Trident(Internet Explorer 7.0) ja Tasman(aiemmat Explorerin versiot).

4.2 Mobiilien laitteiden huomiointi

Sovelluksen suunnittelussa kiinnitettiin huomiota yleistyneisiin mobiileihin laitteisiin ja niiden toiminnallisuuksien huomiointiin, esimerkiksi kosketusnäyttöjen käyttöön. Isoja ongelmakohtia olivat Drag-and-drop, joka toimii, mutta sen käyttöä varten tarvitsi kirjoittaa lisää koodia. Isompi ongelmakohta oli tooltipit, jotka eivät toimi ollenkaan kosketusnäytöllä, koska kosketusnäytöllä joko painetaan näytön ikonia tai ei. Ongelma pyrittiin ratkaisemaan tekemällä ikonit tarpeeksi kuvaaviksi oletuksena, mutta tämä osoittautui haastavaksi tehtäväksi tietyissä paikoissa. Tuotteessa käytettiin paljon standardimaisiksi muodostuneita ikoneja, esimerkiksi T-kirjainta tekstikomponentille ja isoa B-kirjainta tekstin lihavoinnille. Ongelmakohteiksi osoittautuivat lähinnä eri komponentit, joille oli vaikea löytää tietynlaista ikonia, joka kertoisi tarkkaan käyttäjälle sen käyttötarkoituksesta, ja erottaisi sen vastaavista komponenteista. Esimerkkinä tästä mainittakoon vaikka järjestelmä-komponentin ja järjestelmäkomponentti-komponentin erottaminen pelkällä ikonilla.

4.3 Pilvipalvelun valinta

Alkuperäisenä päätöksenä tarkoituksena oli tarjota sovellus SAAS-tyylisenä palveluna, mutta lopulliseen tuote tarjottiin SAAS-tyylisenä sovelluksena, ja yritys piti itse sovellusta Amazonin Elastic- nimisessä IAAS-pilvessä, minkä takia serverien ylläpito oli Amazonin vastuulla. Demon kehityksen aikana sovelluksen toimintaa pilvessä ei mietitty tarkkaan, mutta sovellus yritettiin pitää sellaisena, että se pyörisi lähinnä missä pilvipalvelussa tahansa. Tiimi kartoitti alun perin pari vaihtoehtoista pilvipalvelua, joita silmälläpitäen tehtiin päätöksiä ohjelman suunnittelua ja kielen valintaa koskien. Lopullista palvelunvalintaa ei tehty demon kehityksen aikana.

4.4 Työnkulku

Työn tehtävät kirjattiin Google Docs dokumenttiin, jota käytettiin kaikkeen työn etenemisen seuraamiseen ja taskien kirjaamiseen. Taskien määräys toimi yksinkertaisesti kertomalla toiselle työntekijälle, mitä ominaisuutta oli kehittämässä ja merkitsemällä se dokumenttiin oman nimen alle vapaamuotoisesti. Viikoittaisissa palavereissa käytiin läpi tehdyt ominaisuudet ja suunniteltiin seuraavat työn kohteet. Työnkulku muistutti siis perus AGILE-tyylistä ohjelmistoprosessia, jossa joka viikolle valittiin tiettyjä tehtäviä, joita toteuttaa ja viikoittaisissa kokouksissa katsotaan mitä on tehty ja onko ollut ongelmia. Ainoana poikkeuksena normaaliin työtapaan oli se, että normaalissa AGILE-järjestelmässä työn eteneminen katsotaan päivittäin, mutta koska tiimi toimi pienellä porukalla samassa tilassa, ongelmista pystyttiin kertomaan toisille heti, eikä erillisiä päivittäisiä palavereita tarvittu. Yhteisiä tiedostoja muokattiin lähiverkossa manuaalisesti eli työssä ei käytetty esimerkiksi GIT Bash työkalua. Tiedostojen varmuuskopiointi suoritettiin manuaalisesti viikon välein erilliselle verkkolevylle.

4.5 Razorin valinta

Ohjelmistoprojektin edetessä siihen vaiheeseen, jossa komponenttien ynnä muiden vastaavien valmiiden pakettien lisääminen aloitettiin, koko projekti vaihdettiin käyttämään Microsoftin Razor-kieltä. Lopulta Razorin ominaisuuksia ei käytetty oikeastaan missään, sillä koko ohjelmalla ei ollut tietokantaa, mutta koodista saatiin tehtyä hieman olio-ohjelmointia mukaileva perimällä komponentteja eri dokumentteihin. Saman olisi myös saanut aikaan lataamalla sivuston osat oikeaan paikkaan käyttämällä PHP:n ”load”- metodia.

5 TYÖN ESITTÄMINEN JA TESTAUS

5.1 Komponenttien raahaus

HTML5:n erilaisia ominaisuuksia käytettiin käyttöliittymän perustoimintoihin. Käyttäjä pystyi lisäämään työkalun dokumenttiin erilaisia ennalta määriteltyjä komponentteja, joiden sisältöä pystyi muokkaamaan. Nämä komponentit raahattiin sivun sisällä olevasta laatikosta käyttäjän haluamaan paikkaan dokumentissa. Tämä koko toiminto toteutettiin HTML5:n drag-and-drop metodilla. Drag-and-drop toimii erilaisten eventtien perusteella. Raahauksen aloituksella ja lopetuksella on omat eventtinsä, ondragstart ja ondragend. Raahauksen aikana kuunnellaan omaa eventtiä drag, ja raahattaessa elementtiä tiettyjen elementtien sisään ja niistä ulos, kuunnellaan eventtiä ondragenter ja ondragleave. Pudotus hoituu ondrop metodin kuuntelulla.

5.1.1 Drag

Drag -eventtiä kutsutaan, kun tiettyä elementtiä johon on liitetty ”ondragstart” attribuutti, aletaan raahata. Alla olevassa esimerkissä raahattavan teksti sisältö ja id liitetään omiin muuttujiin myöhempää käyttöä varten.

```
function drag(event)
{
        raahattava_data= event.dataTransfer.setData("Text", event.target.id);
        raahattavan_id = event.target.id;
}
```

5.1.2 Dragenter ja Dragleave

Dragenter-eventtiä kutsutaan kun raahattava kohde siirtyy elementin, jolla on ondragenter-attribuutti, päälle ensimmäisen kerran ja dragleave toimii ondragleave- attribuutin sisältävän elementin kanssa päinvastaiseen suuntaan. Yleensä kaikkiin drag and drop funktioihin tulisi lisätä oletustoiminnon estämisfunktio event.preventDefault(), sillä muuten selain suorittaa erilaisia toimintoja, esimerkiksi kuvaa raahatessa selain avaa kuvan selain-ikkunassa. Tässä tapauksessa käytetään myös event.stopPropagation()-funktiota. Tämä funktio on yksi jQueryn ominaisuuksista ja se estää eventin kuplimasta ylöspäin DOM-puussa, tarkoittaen käytännössä sitä, että kohteen korostus ei vilku päälle ja pois useampaa kertaa. Molemmat alla olevat koodi-esimerkit

käsittelevät objektin korostusta visuaalisesti, kun objekti raahataan komponentin päälle, jolloin käyttäjälle ilmaistaan että pudotuksen tapahtuessa komponentti lisätään tämän komponentin alle.

```
function dragenter(event)
{
    event.preventDefault();
    event.stopPropagation();
    kohde = event.target;
    kohde = $(kohde).closest('.komponentti');
    if( $(kohde).hasClass('drop-highlight') == false )
    {
        $(kohde).addClass('drop-highlight');
    }

}
function dragleave(event)
{
    event.preventDefault();
    event.stopPropagation();
    kohde = event.target;
    kohde = $(kohde).closest('.komponentti');
    if ( $(kohde).hasClass('drop-highlight')  )
    {
        $(kohde).removeClass('drop-highlight');
    }
}
```

5.1.3 Drop

Drop -funktiota kutsutaan kun raahattava objekti pudotetaan elementin päällä, jolla on ondrop attribuutti. Alla olevassa esimerkissä poistetaan kohteen visuaaliset korostukset ja siirytään funktioon, jossa itse komponentin liittäminen oikeaan kohtaan tapahtuu.

```
function drop(event)
{
    event.preventDefault();
    event.stopPropagation();

    if( $(kohde).hasClass('drop-highlight') )
    {
        $(kohde).removeClass('drop-highlight');
    }
    if( $(kohde).hasClass('komponentti') == false )
    {
        kohde = $(kohde).closest('.komponentti').parent();
    }
         prosessoiKentta(raahattavan_id);
}
```

5.1.4 Drop file

Drop file -eventteihin sisältyy tiettyjä ominaisuuksia sen mukaan mitä raahataan. Esimerkiksi tiedostoja raahatessa eventtiin sisältyy muun muassa tiedoston koko ja tiedostotyyppi, kun taas html-elementtejä raahatessa mukana tulee esimerkiksi elementin id:t ja luokat (ks. drop-metodi aiemmin). Aluksi ohjelma tarkisti kohteen tyypin id-attribuutin avulla. Käyttäjän raahatessa komponenttia dokumentin päälle, käyttäjälle annettiin visuaalista palautetta, mihin komponentti olisi putoamassa, mikäli käyttäjä päästäisi irti hiiren napista käyttäen dragenter-eventtiä. Lopulta käyttäjän pudottaessa komponentin dokumenttiin sisälle, ohjelma latasi komponentin rakenteen ja mahdolliset valmiit tiedot ja käyttöliittymän elementit.

```
function dropFile(event) {

    event.stopPropagation();
    event.preventDefault();
    if (event.dataTransfer.files.length < 1)
    {
        drop(event);
    }

    var files = event.dataTransfer.files;
    var length = files.length;
    //tiedoston käsittelyn funktiot tämän alle
…
```

5.1.5 Vaihtoehtoinen metodi

Drag and Drop ominaisuuden tuen puuttuminen muutamasta selaimesta oli iso ongelma tässä toteutuksessa, sillä käyttöliittymän perustoiminnallisuus oli hyvin riippuvainen tästä ominaisuudesta, jolloin ohjelman ei ollut käytettävä tietyissä selaimissa. Lopulliseen versioon komponenttien raahaus toteutettiin jQueryn omalla drag and drop menetelmällä, juurikin yhteensopivuusongelmien takia.

```
    $('.editable').draggable({
        handle: '.dragger',
    });
```

Tähän metodiin liitettiin vielä lisäksi itse komponentin kohtelun sisältävä koodi.

5.2 Tiedoston lisääminen

Kuvan liitäminen tapahtui myös HTML5:den uudella FILE- rajapinnalla joka luki yhden tai useamman tiedoston tiedot.

```
for (var i = 0, f; f = files[i]; i++)
    {
        if (!f.type.match('image.*') )
        {
            console.log('Type match ei ollut image');
            handleFile(files,length);
        }

        var reader = new FileReader();
        reader.onload = (function (theFile) {
            return function (e) {
           // Render thumbnail.
           var span = document.createElement('span');
           span.setAttribute('class', 'thumb-span' + span_counter);
                span_counter++;
                        span.innerHTML = ['<img class="added-picture"
        draggable="false"  src="', e.target.result,
                                  '" title="', escape(theFile.name),
'"/>'].join('');
                $(event.target).append(span);
            };
```

Tällä rajapinnalla pystyi helposti tarkistamaan, että onko tiedosto oikeaa muotoa, esimerkiksi kuva tai Word-dokumentti, tiedoston koon, sekä seuraamaan latausprosessia, joka on hyödyllinen esimerkiksi jos käyttäjälle halutaan antaa visuaalista palautetta siitä, kuinka pitkällä tämän liitteen lisääminen on.

5.3 Komponentin raahaus kosketusnäytöllä

Koska tuotteen haluttiin olevan käytettävissä myös kosketusnäytöllisissä laitteissa, lähinnä mobiililaitteissa, drag-and-drop toteutustapaa piti miettiä tarkemmin. Ongelmana oli lähinnä raahaamisen erottaminen laitteen selaimen, tai laitteen itsensä, kosketuseleistä. Tähän ongelmaan oli ratkaisuna tunnistaa käyttäjän laite, ja määrittää erilaiset kuuntelijat HTML5:den valmiille kosketus-eventeille, sekä tehdä omat metodit jokaiselle raahaus toiminnolle kosketusnäyttöjä varten. Käyttäjälle haluttiin myös antaa mahdollisuus lisätä komponentteja eri tavalla, esimerkiksi painamalla nappia, jolloin komponentti siirtyisi dokumentin loppuun. Kosketusnäytön tuen toteutus jätettiin demon scopen ulkopuolelle, sillä demon esittelytilanne tapahtui normaalilla tietokoneella.

5.4 Työkaluohjeet(tooltips)

Työkaluvinkkejä käytettiin kertomaan käyttäjälle mitä tietyt ikonit tarkoittavat, ja mitä toiminnallisuuksia niiden takana on. Työkaluvinkit toteutettiin käyttämällä HTML5:den valmiita title -attribuutteja, joita muokattiin CSS3:n avulla. Periaatteena tässä oli antaa ikoneille title -attribuutti, jota muokkasimme CSS:llä näyttämään puhekuplalta. Käytännössä tämä piirtäminen tapahtui tekemällä pieni kolmio CSS:n läpinäkyvillä border -attribuuteilla. Toinen mahdollisuus olisi ollut käyttää valmiita scriptejä, mutta päätimme käyttää CSS:ää demossa. Lopulliseen versioon otettiin käyttöön uusin jQuery-versio, jossa tuli mukana valmis tooltip-metodi.

5.5 Ikonit

Työkalussa käytettiin lähinnä valmiiksi tehtyjä ikoneja, jotka saatiin ilmaisia ikoneja tarjoavilta nettisivuilta ja jQueryUI:n mukana tulevista ikonilistoista. Ikonit jätettiin suurimmilta osin tekemättä, sillä niiden suunnittelu ja toteuttaminen olisi vienyt paljon aikaa toiminnallisuuden lisäämiseltä, ja ulkoasun viilausta pidettiin vähemmän tärkeänä itse toiminnallisuuteen verrattuna. Jälkikäteen ajateltuna tiimi olisi ehkä ehtinyt tehdä muutamia ikoneja lähinnä komponentteihin, sillä suurin osa komponentin tekemisestä oli suunnittelua, mikä saattaisi onnistua muun työn ohessa. Valmiiseen tuotteeseen saatiin ikonit ulkoiselta käyttöliittymän suunnittelijalta.

5.5 Testaus

Tuotetta testattiin kahdessa vaiheessa. Jonkin toiminnallisuuden valmistuessa molemmat ohjelmoijat testasivat ominaisuuden mahdollisimman monella eri selaimella ja katsoivat oliko ominaisuus hyvin käytettävä tai voiko sitä mahdollisuus toteuttaa paremmin. Mahdolliset bugit kirjattiin ylös ja niitä yritettiin korjata, mikäli aikaa riitti. Tehty ominaisuus esiteltiin myös viikoittaisissa palavereissa, ja mikäli tuote toimi suunnitellusta poiketen, viat korjattiin ja korjattu ominaisuus esiteltiin uudestaan seuraavassa palaverissa. Lopullisessa tuotteessa käytettiin oikeaa bugien kirjaustyökalua.

6 YHTEENVETO

Työn tavoitteena oli siis tehdä toimiva käyttöliittymä projektin dokumentoinnin hallintaan tarkoitetulle pilvipalvelulle. Työ toteutettiin HTML5, CSS3 ja Javascript kehitystekniikoita käyttäen. Työn tuloksena saatiin toimiva tuotteen toiminnallisuutta ja ulkoasua edustava interaktiivinen demo, jonka avulla tuotetta voitiin kaupata mahdollisille asiakkaille. Demon valmistumisen jälkeen tuotetta alettiin jatkokehittää palvelinpohjaiseksi pilvipalveluksi.

Jälkikäteen ajateltuna monia asioita olisi voitu tehdä eri tavalla. Ohjelman koodissa olisi voinut käyttää huomattavasti enemmän perinnöllisyyttä ja olio-pohjaista toteutustapaa. Esimerkiksi dokumenttiin lisätyt komponentit olivat erillisiä olioita, mutta komponenttien sisäiset yhtenevät ominaisuudet, kuten eri ikonit ja niiden toiminnallisuus, eivät. Tästä johtuen aina kun uusia toiminnallisuuksia lisättiin komponentteihin, jokaista komponenttia jouduttiin muokkaamaan erikseen, ’komponentin otsikko’ -dokumentin muokkauksen sijaan.

Myös komponenttien yläpuolen palkeista olisi voinut tehdä oman templaatin, jota olisi voinut muokata tarpeiden mukaan, ja ladata se kyseisen dokumentin tietynlaisten komponenttien sisälle. Tiettyihin komponentteihin olisi ehkä joutunut tekemään omia toiminnallisuuksia palkkeihin, mutta suurin osa voisi käyttää samanlaisia toimintoja.

Palvelinkielen valinta, ja etenkin sen muutos, vei myös aikaa ilman hyötyä itse projektia ajatellen. Esimerkiksi elementtien latauksen ulkopuolisista tiedostoista olisi voitu toteuttaa helposti PHP:llä. Hukatulla palvelinpuolen koodilla ei kuitenkaan ollut suurta merkitystä, sillä palvelinpuolen ohjelmoitiin valmiissa tuotteessa Djangon avulla.

Käyttöliittymän ulkoasua ajatellen värimaailmaa olisi voinut muokata. Värimaailma valittiin projektin alussa silloisten yrityksen värien perusteella. Koko projektin ajan tiimi oli tyytymätön väreihin niiden antaman pehmeän ja lapsekkaan vaikutelmien takia.. Lopulliseen versioon otettiin käyttöön yksinkertaisempi mustaa ja valkoista käyttävän värimaailma, jossa yrityksen värejä käytettiin lähinnä korostamaan eri elementtejä käyttöliittymästä.

Erilaisten työskentelyä helpottavien ohjelmien, esimerkiksi Git Bash-nimisen ohjelman, käyttöä olisi voinut harkita. Toisaalta uusien ohjelmien, etenkin konsolipohjaisen Git Bashin opetteluun

olisi kulunut ylimääräistä aikaa, ja virheiden mahdollisuus olisi ollut suurempi. Silti siitä olisi ollut suurta apua. Myös esimerkiksi työkalun värimaailman muuttaminen olisi onnistunut helpommin, jos olisi osattu käyttää SASS:ia.

LÄHTEET

Apple 1992. User Experience Guidelines. Hakupäivä 11.5.2014 https://developer.apple.com/library/mac/documentation/userexperience/conceptual/applehiguidelines/HIPrinciples/HIPrinciples.html

CanIUse 2014. Compatability tabels for support of HTML5,CSS3 SVG and more in desktop and mobile browsers,. Hakupäivä 11.5.2014 http://caniuse.com/

Department for Business Innovation & Skills 2010. Guidelines For Managing Projects. Hakupäivä 11.5.2014. https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/31979/10-1257-guidelines-for-managing-projects.pdf

Irish P. & Manian D. .HTML5 & CSS3 Readiness, 2013. Hakupäivä 11.5.2014 www.html5readiness.com

Jobs S. April 2010, Thoughts On Flash. Hakupäivä 11.5.2014 http://www.apple.com/hotnews/thoughts-on-flash/

Johnson J., K 28.7.2010. Tips for designing for Colorblind Users. Hakupäivä 11.5.2014 http://designshack.net/articles/accessibility/tips-for-designing-for-colorblind-users/

jQuery 2014, Browser Support. Hakupäivä 11.5.2014 http://jquery.com/browser-support/

StatCounter. Hakupäivä 11.5.2014 http://gs.statcounter.com/#desktop+mobile-comparison-ww-monthly-201112-201312

United States Computer Emergency Readiness Team 2011. The Basics of Cloud Computing. Hakupäivä 11.5.2014, http://www.us-cert.gov/sites/default/files/publications/CloudComputingHuthCebula.pdf

W3C 2014. HTML5 A vocabulary and associated APIs for HTML and XHTML. Hakupäivä 11.5.2014
http://www.w3.org/TR/html5/